sevenstax Trivial File Transfer Protocol

Specification and User Manual

Revision No.:	2.1
State:	Released
Author:	sevenstax GmbH

Initial version:	21/08/07
Last change:	14/01/08
Last Review:	14/01/08
Publication:	Public
Filename:	sevenstaxTFTP_UserManual_v21

Copyright (c) 2008 by sevenstax GmbH
This document is an intellectual property of sevenstax GmbH. Unauthorized copying and distribution is prohibited.

Table of Contents

1 Introduction 3

2 Using TFTP 3

2.1 User modes 3
2.2 Initialization 3
2.3 Using as Client 4
2.4 Using as Server 4
2.5 Using as Client + Server 4

3 Internal Data Structures 4

3.1 TFTP_DATA_TYPE structure 4

4 Public Functions 5

4.1 stxTFTP_Init (handler) 5
4.2 stxTFTP_Start (ServerClient) 5
4.3 stxTFTP_Stop ( ) 6
4.4 stxTFTP_Tick ( ) 6
4.5 stxTFTP_WriteData (remote_ip_address, szFilename, eTFTPType) 7
4.6 stxTFTP_Read (remote_ip_address, szFilename, eTFTPType) 7

5 Notifycodes 9

5.1 NC_TFTP_GETDATA (TFTP_DATA_TYPE, block-number) 9
5.2 NC_TFTP_RECV (TFTP_DATA_TYPE, block-number) 9
5.3 NC_TFTP_REQ_WRITE (TFTP_DATA_TYPE, TransferMode) 10
5.4 NC_TFTP_REQ_READ (TFTP_DATA_TYPE, TransferMode) 10
5.5 NC_TFTP_RX_READY 10
5.6 NC_TFTP_RX_ERROR 11
5.7 NC_TFTP_TX_READY 11
5.8 NC_TFTP_TX_ERROR 11

6 Adjustable Parameters 11

7 Restrictions 12

8 Change History 13

1 Introduction

TFTP is a very simple Protocol used to transfer files via UDP. sevenstaxTFTP is part of inside the *sevenstax Internet Protocols Suite*.

It can be enabled via *#define TFTP_SUPPORTED = 1* in features.h. The TFTP protocol works ahead UDP, so *UDP_SUPPORTED = 1* must be set too.

For basic information on how to use the *sevenstax Internet Protocols Suite,* please refer to the *sevenstaxTCP/IP Users Manual*.

2 Using TFTP

The Trivial File Transfer Protocol is used to transmit data between a client and a server. It is a very simple and small protocol and therefore it can easily be implemented in embedded systems with small resources.

2.1 User modes

The module works in three different modes:

- Client-Mode
- Server-Mode
- Client+Server-Mode

In Client-Mode the device is able to send Read-Requests (RRQ) and Write-Requests (WRQ), but it can't receive them. The device sends the request from a random UDP-Port.

In Server-Mode the device is able to answer RRQ and WRQ. The default port to receive requests is UDP-Port 69.

In Client + Server-Mode it is possible to do both, sending and receiving RRQ and WRQ. The UDP-Port to receive the request is the same as in Server-Mode and the port to send them is also random as in Client-Mode.

To use TFTP with sevenstaxTCP/IP you should proceed the following way (please refer to the following chapters for details):

2.2 Initialization

- #define *TFTP_SUPPORTED =1* in 'features.h'. Also UDP_SUPPORTED must be enabled there.
- The application has to prepare a dedicated callback function, to react Notify Codes in of the TFTP module.
- Call *stxTFTP_Init()* once to set-up sevenstaxTFTP's internal state and to register the applications callback function.

- In your main loop or operating system task call *stxTFTP_Tick()* repeatedly to keep it alive.

2.3 Using as Client

- To start the module in Client-Mode, call: *stxTFTP_Start(TFTP_MODE_CLIENT).* Now the Module is able to send Requests.
- To get a file from a remote TFTP server, use the function *stxTFTP_ReadData( ).*
- To transmit data to a TFTP server, use the function *stxTFTP_WriteData( ).*

2.4 Using as Server

- To start the module in Server-Mode call: *stxTFTP_Start(*TFTP_MODE_SERVER). Now the Module is able to receive Requests (RRQ and WRQ) on the defined Port (default: 69).
- The server will automatically respond to remote read / write requests.
- The server will automatically call the user application, if it requires or delivers any user data.

2.5 Using as Client + Server

- To start the module in Client-Server-Mode call: *stxTFTP_Start(* TFTP_MODE_CLIENTSERVER)
- In this mode the module is able to receive on the defined port and to send on a random port.

3 Internal Data Structures

3.1 TFTP_DATA_TYPE structure

The TFTP_DATA_TYPE structure is used to hold information about incoming and outgoing data packets.

It delivers information from application to the TFTP module about of the data to be send. It delivers information from the TFTP module to the application about of the data to being received.

```
typedef struct _tag_tftp_senddata
{
  IPV4 FPTR_stx        dest_ip_address;                    // IP-Address of the remote
  CHAR_stx             szFilename[TFTP_MAX_FILENAME_SIZE]; // Filename of RRQ or WRQ
  UINT8_stx FPTR_stx   fpData;                             // Pointer to the data
  UINT16_stx           wSize;                              // data size (max. 512 bytes)
} TFTP_DATA_TYPE;
```

IPV4 FPTR_stx dest_ip_address

Pointer to an IPV4 address structure containing the IP-Address of the remote peer. Please refer to the “sevenstaxTCP/IP Users Manual” for details.

CHAR_stx szFilename [TFTP_MAX_FILENAME_SIZE]

File name of RRQ or WRQ.

FPTR_stx fpData

Pointer to the data delivered / received.

UINT16_stx wSize

Amount of data in bytes (max. 512 bytes, <512 means 'End Of File').

4 Public Functions

4.1 stxTFTP_Init (handler)

Description

Set up some internal variables and registers the application callback function. It must be called once before using any other TFTP functions.

Parameter

PROTOCOL_NOTIFY_HANDLER handler
Application handler function to receive notify codes from sevenstaxTFTP module. The handler function will be called, when TFTP data is received, TFTP data should be send, to inform the application about an error or that a transfer has been finished successful.

Return Value

void

Comment

-

4.2 stxTFTP_Start (ServerClient)

Description

This function starts the module in “Server”, “Client” or “Client + Server” mode, depending on the delivered parameter.

Parameter

TFTP_MODE ServerClient

Via Parameter *ServerClient* TFTP module starts in different modes:

If *ServerClient* is TFTP_MODE_CLIENT, the device is able to get data from or put data to a TFTP server.

If *ServerClient* is TFTP_MODE_SERVER, the device is able to receive data from or submit data to clients.

If *ServerClient* is TFTP_MODE_CLIENTSERVER, the device is server and client.

Return Value

void

Comment

stxTFTP_Init() must have been called before using this function.

Please note, that the module supports only one TFTP session at a time, a server OR client session. The session first must be closed, before switching to the other mode.

4.3 stxTFTP_Stop ( )

Description

This function stops the current mode of the device, therefore data can´t be exchanged anymore.

Parameter

void

Return Value

void

Comment

stxTFTP_Start() must have been called before using this function.

Please note, that there is no need stop the session, to switch between server and client mode, if it was initialized with TFTP_MODE_CLIENTSERVER mode.

4.4 stxTFTP_Tick ( )

Description

This function keeps TFTP alive. It should repeatedly be called by the application, like all other sevenstax “Tick” functions too, independently of TFTP is stopped or not.

Parameter

void

Return Value

void

Comment

This function will not consume application runtime, if no TFTP session is active.

4.5 stxTFTP_WriteData (remote_ip_address, szFilename, eTFTPType)

Description

Forces the TFTP-module to send a Write-Request to a remote TFTP-Server. If the request is accepted, the notify-handler will be called with the notifycode NC_TFTP_GETDATA.

Parameter

IPV4 remote_ip_address
Target's IP-Address

STRING_stx szFilename
Case sensitive filename used by the opponent to store the transmitted data.

UINT8_stx eTFTPType
Type of transfer mode: TFTP_TRFM_OCTET only (TFTP_TRFM_NETASCII not yet supported)

Return Value

BOOL_stx

TRUE_stx	if Request has been sent
FALSE_stx	if any error occurs

Comment

stxTFTP_Start() must have been called with *TFTP_MODE_CLIENT* or *TFTP_MODE_CLIENTSERVER* before using this function. It is not possible to send a write-request in Server-Mode (TFTP_MODE_SERVER).

Please note, that the file name will be handled case sensitive!

4.6 stxTFTP_Read (remote_ip_address, szFilename, eTFTPType)

Description

Forces the TFTP module to send a Read-Request to a remote TFTP-Server.
If the request is accepted, the notify-handler will be called with the notifycode NC_TFTP_RECV.

Parameter

IPV4 remote_ip_address
Target's IP-Address.

STRING_stx szFilename
Case sensitive name of the file requested from the remote server.

UINT8_stx eTFTPType
Type of transfer mode: TFTP_TRFM_OCTET only (TFTP_TRFM_NETASCII not yet supported)

Return Value

BOOL_stx

TRUE_stx	if Request has been sent
FALSE_stx	if any error occurs

Comment

stxTFTP_Start( ) must have been called with TFTP_MODE_CLIENT or TFTP_MODE_CLIENTSERVER before using this function. It is not possible to sends a read-request in Server-Mode (TFTP_MODE_SERVER).

Please note, that the file name will be handled case sensitive!

5 Notifycodes

This is a list of notifycodes generated by sevenstaxTFTP.

Notifycodes inform the user application about any event generated by a sevenstax module. For details regarding the use of the notify-handler, please refer to the “sevenstaxTCP/IP Users Manual“.

5.1 NC_TFTP_GETDATA (TFTP_DATA_TYPE, block-number)

This event is generated by the TFTP module to get a user data from the application, to be sent as the next data block.

Therefore, the application must set the pointer to the beginning of the next 0..(max.) 512 bytes block of data and set the data size.

The same block might be requested multiple times, if the TFTP module detected errors (e.g. lost packets). That´s why it is necessary to observe the block-number. The datapointer position can be calculated by block-number * 512.

Parameter a points to a TFTP_DATA_TYPE structure, which contains the remote IP-Address, the requested file name, a data pointer and the data size. The pointer must be set to the expected data and the data size must be set to the right size (maximal 512 bytes) .

Parameter b indicates the block number of the data, which is requested.

ReturnValue indicates if the application result:

TFTP_ERR_OK	if accepted by application
TFTP_ERR_xxx	if any application problem

A TFTP_ERR_xxx will result in sending the adequate TFTP error message to the remote station.

5.2 NC_TFTP_RECV (TFTP_DATA_TYPE, block-number)

This event is generated by the TFTP module to deliver the next data block to the user application.

The TFTP_DATA_TYPE includes a pointer to the received data-block. The block number indicates the position of the received data. The file offset position can be calculated by block-number * 512.

Parameter a points to the TFTP_DATA_TYPE structure. The structure contains the remote-IP-Address, the file-name, a datapointer and the data-size. The datapointer points to the received data and the size of the data is given by the data-size.

Parameter b indicates the block number of the received data.

ReturnValue indicates the application result:

TFTP_ERR_OK	if accepted by application
TFTP_ERR_DISKFULL	if any application error

5.3 NC_TFTP_REQ_WRITE (TFTP_DATA_TYPE, TransferMode)

This event notifies the application for an incoming Write-Request.

The TFTP_DATA_TYPE includes the filename to write (only in Server- or Client-Server-Mode) and the remote-IP-Address. The application has to decide, weather to accept the request or not.

Parameter a points to a TFTP_DATA_TYPE structure, which contains the remote IP-Address and the file name. The data pointer and the data size is not used by this event.

Parameter b indicates the transfer mode (octett or netascii). Netascii is not supported in this version.

ReturnValue indicates the application result:

TFTP_ERR_OK

TFTP_ERR_ACCESS_VIOLATION

TFTP_ERR_DISKFULL

TFTP_ERR_ILLEGAL_OPERATION

TFTP_ERR_FILE_ALREADY_EXISTS

Please note, that any authentication or access right management might be solved by the user application at this time.

5.4 NC_TFTP_REQ_READ (TFTP_DATA_TYPE, TransferMode)

This event notifies the application of an incoming Read-Request.

The TFTP_DATA_TYPE includes the name of the file requested by the client (only in Server- or Client-Server-Mode).

Parameter a points to a TFTP_DATA_TYPE structure, which contains the remote IP-Address and the filename. The datapointer and the datasize is not used by this event.

Parameter b indicates the transfer mode (octet or netascii). netascii is not supported in this version.

ReturnValue indicates the application result:

TFTP_ERR_OK

TFTP_ERR_FILE_NOT_FOUND

TFTP_ERR_ACCESS_VIOLATION

TFTP_ERR_ILLEGAL_OPERATION

5.5 NC_TFTP_RX_READY

This event notifies the application that the current receive transfer finished successfully.

Parameter a, b and ReturnValue are not used.

5.6 NC_TFTP_RX_ERROR

This event notifies the application that the current receive transfer was aborted with an error.

Parameter a, b and ReturnValue are not used.

5.7 NC_TFTP_TX_READY

This event notifies the application that the current transmit transfer finished successfully.

Parameter a, b and ReturnValue are not used.

5.8 NC_TFTP_TX_ERROR

This event notifies the application that the current transmit transfer was aborted with an error.

Parameter a, b and ReturnValue are not used.

6 Adjustable Parameters

TFTP parameters are defined in '*tftpdefs.h*'. These parameters should properly be adjusted depending on the application requirements. The following list describes them in detail.

TFTP_SERVERPORT_LOCAL **default: 69** **valid: 1..n**

RFC conform UDP-Port to listen for requests (for server-mode/client-server-mode). Might be adopted to special user requirements.

TFTP_SERVERPORT_REMOTE **default: 69** **valid: 1..n**

RFC conform Remote UDP-Port to connect to (for client-mode/client-server-mode). Might be adopted to special user requirements.

TFTP_MAX_FILENAME_SIZE **default: 60** **valid: 1..n**

Buffer size to handle filenames of read/write requests. Might be adopted to users file system or memory requirements.

7 Restrictions

As sevenstaxTFTP is developed for very tiny embedded applications, there are a few restrictions, which must considered. The restrictions result from the requirement, to run on very small devices with a small amount of RAM/ROM resources:

Only one TFTP session at a time

sevenstaxTFTP can only be used for one single session, as client or server.

Further remote client connection trials will silently be ignored. User application tries to open a client session while a session already is active, will return error code.

'octet' / 'binary' mode supported only

sevenstaxTFTP currently supports 'octet' (might also be called 'binary') mode only. This is a raw binary transfer, which can be used for ASCII files too. Using the 'octet' mode helps to avoid target system specific conversion problems.

Receiving 'netascii' read/write request by the server will result in sending error messages to the remote client. User application calls using the 'netascii' will result in error return codes.

8 Change History

Ver.	*Date*	*by*	*Change description*
1.0	21-08-2007	bbr	base version
2.0	08-01-2008	ttu	review, source-code changes considered
2.1	14-02-2008	krz	review

The information furnished in this document is believed to be accurate and reliable. However, no responsibility is assumed by sevenstax for its use, nor for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent rights of sevenstax.

This document is an intellectual property of sevenstax GmbH. Unauthorized copying and distribution is prohibited.

Copyright (c) 2008 by sevenstax GmbH